Cluster By: A New SQL Extension for Spatial Data Aggregation*

Chengyang Zhang, Yan Huang
Computer Science and Engineering Department
University of North Texas
Denton, TX 76201
[chengyang,huangyan]@unt.edu

ABSTRACT

The development of areas such as remote and airborne sensing, location based services, and geosensor networks enables the collection of large volumes of spatial data. These datasets necessitate the wide application of spatial databases. Queries on these geo-referenced data often require the aggregation of isolated data points to form spatial clusters and obtain properties of the clusters. However, current SQL standard does not provide an effective way to form and query spatial clusters. In this paper, we aim at introducing *Cluster By* into spatial databases to allow a broad range of interesting queries to be posted on spatial clusters. We also provide a language construct to specify spatial clustering algorithms. The extension is demonstrated with several motivating examples.

Categories and Subject Descriptors

H.2.3 [**Database Management**]: Languages—*Query languages*; H.2.3 [**Database Management**]: Database Applications—*Data mining*; H.2.3 [**Database Management**]: Database Applications—*Spatial databases and GIS*

General Terms

Languages

Keywords

SQL, spatial clustering, spatial databases, GIS

1. INTRODUCTION

The development of areas such as remote and airborne sensing, location based services, and geosensor networks enables the collection of large volumes of spatial data in increasingly finer resolution. These ever growing datasets necessitate the wide application of spatial databases [15, 17, 13]. Queries on these geo-referenced data often require the aggregation of isolated data points to form spatial clusters and obtain properties of the clusters. However, current SQL standard does not provide an effective way to form and query spatial clusters. For example, without sophisticated programming efforts, it is difficult to find the length of a traffic jam using speed data collected by loop detectors deployed along highways. In this paper, we aim at introducing *Cluster By* into spatial databases to allow a broad range of interesting queries to be posted on spatial clusters.

Example 1(Motivating Scenario: Community Analysis) Agencies such as National Center for Education Statistics may want to identify low income neighborhoods- clusters of neighboring houses with low household income. Other than visualization, it may also be interesting to query the area of each low income neighborhood, or find school districts near each of the area. Let's assume a spatial database is used to store the house information using a table called *House*. The *House* table includes the following information: *house location*(represented as spatial object - points), *address, owner name*, and *household income*. How do we find the area of each low income neighborhood? What about finding out all the schools within 3 miles of each low income area if there is another table called *School* that contains all the school locations? One may think of using the following statement for the first question: (Assuming using PostGIS spatial database[1])

```
SELECT    ST_AREA(House.location)
FROM      House
WHERE     House.household_income < 30K
GROUP BY  House.location
```

However, the above statement will not work for two reasons. First of all, semantically *Group By* simply classifies tuples with the ***same*** values into one group. However, a spatial cluster typically consists of ***nearby*** locations instead of ***same*** locations and each house has a ***distinct*** location. Even if *Group By* was generalized into a *Cluster By* like concept, still the spatial function *ST_AREA* can not have *House.location* as an operand since *House.location* represents a set of locations, not a spatial object such as polygon. Similarly, we can not answer the second question regarding

*This work was partially supported by the Texas Advanced Research Program under Grant No. 003594-0010-2006 and by the National Science Foundation under Grant No. CNS-0709285.

ACMGIS'07, November 7-9, 2007, Seattle, WA

[1]The spatial functions in this paper are all from PostGIS - the spatial cartridge of PostgreSQL database. Their counterparts in other spatial databases may have slightly different names and syntaxes.

the schools near each low income neighborhood because the spatial geometry can not be abstracted from discrete point data.

Our solution to the problem is to extend current SQL language construct to:

- allow spatial data points with similar non-spatial property, e.g. household income, to be clustered according to spatial ***proximity*** and then a cluster of points may be collectively treated as a spatial object, such as a polygon.
- delegate spatial DBMS to process and optimize related queries intelligently: find the spatial clusters, elevate them into spatial objects, and support queries on these objects.

This paper provides the first step, i.e., defining the basic syntax of the *Cluster By* extension for SQL towards solving this problem, while leaving the query processing and implementation issues to our future work. The rest of the paper is organized as follows. Section 2 summarizes related work. The usage of *Cluster By*, including the specification of spatial clustering algorithm, is illustrated in section 3. Finally, section 4 concludes the paper and discusses our future work.

2. RELATED WORK

Our proposed extension, *Cluster By*, stems from the inadequacy of traditional *Group by* in spatial data aggregation, as well as the demands for supporting spatial clustering through the database query languages in many applications.

In traditional SQL[6]-compliant database, *Group By* is the main aggregation mechanism to group individual tuples with the ***same*** grouping attribute(s) values together and form one tuple. Spatial database systems build on traditional database systems as cartridges and support spatial data types and predicates, such as those specified by Open GIS simple feature specification [11]. Therefore, *Group By* can also be used for spatial database queries. However, *Group By* point locations is almost meaningless given each spatial data point usually has a ***distinct*** location.

Spatial clustering is a well studied area in data mining. Various clustering models have been proposed, including partition-based clustering[8, 10], hierarchical clustering[19, 5], density-based clustering[4, 3], grid-based clustering[16] etc. However, supporting spatial clustering through spatial database and query language, has not be explored.

Our work is also related to spatial amalgamation [2, 12, 20] in the context of spatial data mining and data warehousing. Spatial amalgamation originated from digital cartography and refers to the process of combing low level spatial objects to form higher level spatial objects. The extensions proposed by [2] group spatial objects with similar non-spatial properties with respect to their location. However the grouping is intended for visualization. Efficient algorithms have been proposed in [12, 20] to aggregate spatial objects. However, neither intuitive language extension to SQL nor efficient query processing algorithms were proposed to integrate amalgamation into spatial database systems.

Our work is further related to sensor database systems due to the inherent georeferenced nature of data collected by sensor networks. Several pioneering works and research prototypes [1, 9, 18, 14] have demonstrated the feasibility of representing and querying a sensor network as a distributed database system. Users can interact with the network using declarative database query languages. However, all these work treated sensors as points with isolated geo-locations generating data that maybe joined with data generated by other sensors. Therefore without considerable programming efforts, it is very difficult to query on aggregations of sensors. By contrast, our work focuses on viewing a collection of sensor readings as a spatial cluster whose properties, e.g. area, can be further queried or joined with other spatial objects.

To the best of our knowledge, the only work (concurrently and independently developed from ours) that recognizes the inadequateness of aggregation using *Group By* in SQL, appears in a very recent paper[7] on generalizing *Order By* and *Group By* to support fuzzy grouping and top-k ranking. This work proposed to extend *Group By* into *Group By* $c_1, c_2, \ldots, c_m$ *into* t with the semantics of clustering tuples into t groups according to attributes $c_1, c_2, \ldots, c_m$. By contrast, our work focuses on extending language construct to allow amalgamation and elevation of discrete spatial data into spatial objects, and support queries on them.

3. CLUSTER BY OPERATORS

In this section, we will use several motivating examples to demonstrate the usage of *CLUSTER BY* clause in different types of spatial aggregation queries.

The syntax for the *Cluster By* is:

```
SELECT          cluster_aggregate_function
                (spatial_attribute)
                [,aggregate_function
                (non_spatial_attribute(s))]
FROM            table_name
WHERE           condition
CLUSTER BY      spatial_attribute
[ALGORITHM      algorithm_name(para1, para2, ...)]
```

There are two major differences from original SQL's *Group By* clause, in addition to the semantic difference between *Group By* and *Cluster By*:

- The cluster_aggregate_function accepts spatial attribute(s) as operand(s) and aggregates each cluster into a spatial object such as a polygon or a line.
- we also provide an (optional) *ALGORITHM* clause immediately after the *Cluster By* to allow the specification of clustering algorithms. This clause can be omitted, which indicates the DBMS to use the default clustering algorithm.

Example 2(Community Analysis: Query Illustration) Continuing with the motivating example in Section 1, the first question regarding the size of each low income neighborhood can now be answered with the following statement:

```
SELECT      ST_Area(ST_Polygon(House.location))
FROM        House
WHERE       House.household_income < 30K
CLUSTER BY  House.location
```

ST_Polygon is a new cluster aggregation function that elevates a set of points to a polygon and can be omitted as a short-hand if the context is clear.

Figure 1 shows intuitively how the queries are processed in a spatial database with the *Cluster By* extension[2].

Note that the *Algorithm* clause is omitted in the above query. Therefore the system uses the default clustering algorithm, such as the DBSCAN[4] algorithm. However, clustering algorithm is independent of the proposed extension and it is up to the database designer to choose the default clustering algorithm.

Similarly, the second question regarding the schools near each low income neighborhood can be answered with the following statement(Note that a subquery is required to perform the join operation involving the clusters):

```
SELECT      School.name,
FROM        School,
           (SELECT      ST_Polygon(House.location)
            AS          lowIncomeNeighborhood
            FROM        House
            WHERE       House.household_income < 30K
            CLUSTER BY  House.location
            )
WHERE       ST_Distance(school.location,
            lowIncomeNeighborhood)<=3
```

Another source of large volumes of geo-referenced point data is sensors. With the proposed *Cluster By*, discrete sensor data can now form spatial clusters and be queried about elegantly, which will be further demonstrated with following examples.

Example 3(Traffic Jam Detection) Find the line segments and their lengths of I-35 that is blocked by traffic jam. The traffic jam is determined by vehicle speed sensors monitored at discrete points on highways.

```
SELECT        ST_Line(speedSensor.location),
              ST_Length(ST_Line(speedSensor.location))
FROM          speedSensor, highway
WHERE         speedSensor.speed < 20
AND           highway.name = "I-35"
AND           ST_Contains(highway.geom,
              speedSensor.location)=TRUE
CLUSTER BY    speedSensor.location
```

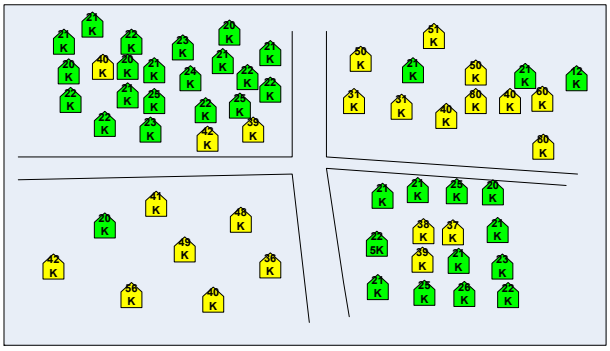
ST_LINE is a new aggregation function that elevates a set of points to a line segment.

Example 4(Flooding Damage Estimation) During and after a flooding, the city hydrologic engineering center needs to analyze the area and damage of the flooding by listing all the houses inside the flooding region. The flooding region is monitored by a network of water level sensors. A location is considered as flooded if the surface water level is above d. A spatial database is used to store the house information. This time user wants to use DBSCAN algorithm to find clusters. Let Eps denote the threshold of the distance, $MinPts$ denotes the threshold of the number of points[3].

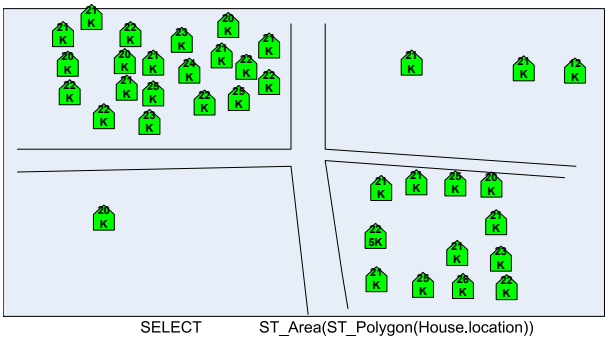
```
SELECT      House.address,
            House.location,
FROM        House,
            (SELECT      ST_Polygon(WaterLevel.location)
             AS          floodingRegion
```

[2]The numbers marked on the house represent household income. These are synthetic data for illustration use only.

[3]The details of DBSCAN clustering algorithm can be found in [4].

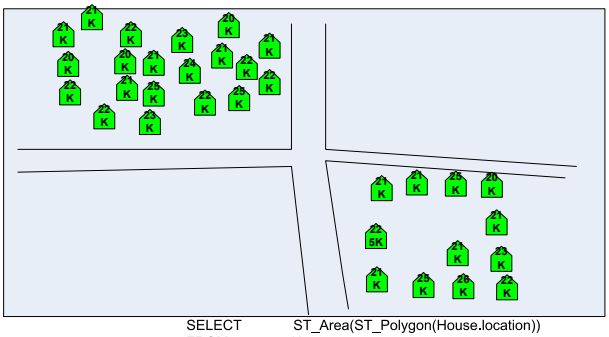


(a) Initial Data



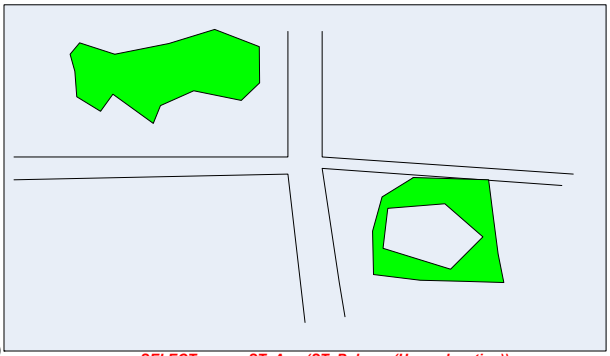
```
SELECT      ST_Area(ST_Polygon(House.location))
FROM        House
WHERE       House.household_income < 30K
CLUSTER BY  House.location
```

(b) Where Condition



```
SELECT      ST_Area(ST_Polygon(House.location))
FROM        House
WHERE       House.household_income < 30K
CLUSTER BY  House.location
```

(c) Clustering



```
SELECT      ST_Area(ST_Polygon(House.location))
FROM        House
WHERE       House.household_income < 30K
CLUSTER BY  House.location
```

(d) Aggregation

Figure 1: Query Illustration

```
            FROM        WaterLevel
            WHERE       WaterLevel.waterDepth > 10
            CLUSTER BY  WaterLevel.location
            ALGORITHM   DBSCAN(Eps,MinPts))
WHERE       ST_Contains(floodingRegion,
            House.location)=TRUE
```

In addition to the ones used above, various spatial functions and predicates can be used with the *Cluster By* clause after a cluster is elevated into a polygon or line. Table 1 gives a list of a few more functions and predicates that may be supported once clusters are formed and elevated into spatial objects.

Spatial functions	ST_Distance, ST_Centroid, ST_Length, ST_Area, ST_Boundary, ST_Union, ST_Difference etc.
Spatial Predicates	ST_Within, ST_DWithin, ST_Equals,ST_Disjoint, ST_Intersects, ST_Touches, ST_Crosses, ST_Overlaps, ST_Contains, ST_Covers, ST_CoveredBy etc.

Table 1: Spatial Aggregates and Predicates

4. SUMMARY

In this paper we have proposed a new extension of SQL - *Cluster By* for spatial data aggregation. Examples are given to demonstrate its effectiveness in expressing queries on spatial clusters. We also provide a language construct to specify the clustering algorithms. Currently we are looking into the implementation of the extension in PostgreSQL with its spatial cartridge - PostGIS.

Our future work includes extending the *Cluster By* to temporal data, spatio-temporal data and streaming data. Query processing is another issue that needs to be addressed for both static and dynamic scenarios. Finding spatial clusters is, in general, more expensive compared with evaluating boolean conditions, or joins. This will become a prominent problem when streaming data is involved. Incremental clustering and spatial indexing mechanisms therefore should be adapted and implemented to support the new proposed extension efficiently. Finally, supporting spatial interpolation through the database may also be useful when the sampling data points are too sparse to represent the monitored spatial phenomena.

5. REFERENCES

[1] P. Bonnet, J. Gehrke, and P. Seshadri. Towards sensor database systems. In *MDM '01*, pages 3–14, 2001.

[2] M. Ester, H.-P. Kriegel, and J. Sander. Spatial data mining: A database approach. In *SSD '97*, pages 47–66, 1997.

[3] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, and X. Xu. Incremental clustering for mining in a data warehousing environment. In *VLDB '98*, pages 323–333, 1998.

[4] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD '96*, pages 226–231, 1996.

[5] S. Guha, R. Rastogi, and K. Shim. Cure: an efficient clustering algorithm for large databases. In *SIGMOD '98*, pages 73–84, 1998.

[6] ISO/IEC 9075. Database language sql, international standard., 1992.

[7] C. Li, M. Wang, L. Lim, H. Wang, and K. C.-C. Chang. Supporting ranking and clustering as generalized order-by and group-by. In *SIGMOD '07*, pages 127–138, 2007.

[8] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[9] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. In *SIGMOD '03*, pages 491–502, 2003.

[10] R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *VLDB '94*, pages 144–155, 1994.

[11] Open GIS Consortium. Open GIS Simple Features Specification for SQL Revision. `http://www.opengis.org/public/sfr1/sfsqlrev10.pdf`, 1998.

[12] S. Prasher and X. Zhou. Multiresolution amalgamation: dynamic spatial data cube generation. In *ADC '04*, pages 103–111, 2004.

[13] P. Rigaux, M. Scholl, and A. Voisard. *Spatial databases with application to GIS*. Morgan Kaufmann Publishers Inc., 2002.

[14] M. Sharifzadeh and C. Shahabi. Supporting spatial aggregation in sensor network databases. In *GIS '04*, pages 166–175, 2004.

[15] S. Shekhar and S. Chawla. *Spatial Databases: A Tour*. Prentice Hall, 2003.

[16] W. Wang, J. Yang, and R. R. Muntz. Sting: A statistical information grid approach to spatial data mining. In *VLDB '97*, pages 186–195, 1997.

[17] M. Worboys and M. Duckham. *Geographic Information Systems: A Computing Perspective (2nd Edition)*. CRC Press, 2004.

[18] Y. Yao and J. Gehrke. The cougar approach to in-network query processing in sensor networks. *SIGMOD Rec.*, 31(3):9–18, 2002.

[19] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: A new data clustering algorithm and its applications. *Data Min. Knowl. Discov.*, 1(2):141–182, 1997.

[20] X. Zhou, D. Truffet, and J. Han. Efficient polygon amalgamation methods for spatial olap and spatial data mining. In *SSD '99*, pages 167–187, 1999.